Optimal Online Scheduling with Arbitrary Hard Deadlines in Multihop Communication Networks

Zhoujia Mao, C. Emre Koksal, Ness B. Shroff
E-mail: maoz@ece.osu.edu, koksal@ece.osu.edu, shroff@ece.osu.edu

***Abstract*—The problem of online packet scheduling with hard deadlines has been studied extensively in the single hop setting, whereas it is notoriously difficult in the multihop setting. This difficulty stems from the fact that packet scheduling decisions at each hop influences and are influenced by decisions on other hops and only a few provably efficient online scheduling algorithms exist in the multihop setting. We consider a multihop wired network (interference free and full duplex transmissions) in which packets with various deadlines and weights arrive at and are destined to different nodes through given routes. We study the problem of joint admission control and packet scheduling in order to maximize the cumulative weights of the packets that reach their destinations within their deadlines. We first focus on uplink transmissions in the tree topology and show that the well known earliest deadline first algorithm achieves the same performance as the optimal off-line algorithm for any feasible arrival pattern. We then address the general topology with multiple source-destination pairs, develop a simple online algorithm and show that it is $O(P_M \log P_M)$-competitive under mild conditions where P_M is the maximum route length among all packets. Our algorithm only requires information along the route of each packet and our result is valid for general arrival samples. Moreover, we show that $O(P_M \log P_M)$-competitive is the best any online algorithm can do. Via numerical results, we also show that our algorithm achieves performance that is comparable to the non-causal optimal off-line algorithm. To the best of our knowledge, this is the first algorithm with a provable (based on a sample-path construction) competitive ratio, subject to hard deadline constraints for general network topologies.**

I. Introduction

We consider a multihop wired network in which nodes receive packets with various (hard) deadlines, enqueued at the intermediate nodes through multiple hops along given routes to given destinations. We assume a time slotted system in which each packet has an identical (unit) length and each link in the network can serve an integer number of packets at a given time slot. Each packet has a certain weight and a deadline and we address the problem of scheduler design in order to maximize the total weight over the packets that are successfully transferred to their destinations within their deadlines. We first focus on the tree topology and show that the Earliest Deadline First (EDF) algorithm achieves the same performance as the optimal off-line algorithm for any feasible or *under-loaded* network arrival pattern[1]. Next, we study the general topology with multiple source-destination pairs. We develop a low-complexity on-line joint admission control and packet scheduling scheme and evaluate its competitive ratio with respect to the cumulative weight achieved by the optimal off-line algorithm. Our scheme only requires information of the packet queues along the route of each packet and is competitively optimal among all online algorithms. To the best of our knowledge, this is the first scheme with a provable (based on a sample-path construction) competitive ratio in general network topologies.

The on-line packet scheduling problem with hard deadlines is gaining increasing importance with the emergence of cloud computing, large data centers, and grid communications. In such applications, a large amount of time-sensitive information needs to be carried among servers and users over a mainly-wired infrastructure. Meeting the deadline requirements of these packets with an efficient use of resources requires a careful design of schedulers that decide on how and when data should be transferred over the network. Due to the large volume of data, the complexity of schedulers should be kept low to reduce the amount of energy consumed by these data centers. To that end, our objective is to develop a low-complexity and provably efficient scheduler and an associated admission controller for deadline-constrained data.

On-line packet scheduling has been a widely-studied problem. Since the seminal work in [1], various versions of the problem for single hop systems have been considered. It has been shown that EDF has the same performance as the optimal off-line algorithm [1, 2] for the scenario in which the system is under-loaded. When considering over-loaded arrivals (i.e., the case when even the best off-line policy drops some packets), there is the additional question of whether the controller needs to decide to accept or reject a packet upon arrival time, i.e., admission control. With the constraint that the admission controller and the scheduler do not have to decide on a packet's admission into the system and the period that it is scheduled upon arrival time, it is shown in [3] that $\frac{1}{4}$ is the best competitive ratio among all on-line algorithms and an online algorithm is provided [4] to achieve this ratio. With this constraint the problem is addressed in [5, 6]. In addition to immediate decisions, the model studied in [7] imposes a penalty on the unfinished workload, and the authors propose an on-line algorithm with competitive ratio $3 - 2\sqrt{2}$ and show that this ratio is the best achievable ratio for all on-line algorithms. Within the single hop setting, similar problems of

[1]Note that a network arrival pattern is said to be under-loaded if there exists an off-line algorithm under which all packets can be served before their deadlines.

packet scheduling have been studied in [8–12] for the scenario with parallel processors, where the controller needs to decide which machine to process each packet as well as scheduling and admission control. An on-line algorithm requiring immediate decision upon packet arrival time is proposed in [9] with an asymptotic competitive ratio $\frac{e-1}{e}$. It is later shown in [10] that this ratio is the maximum achievable ratio for any on-line algorithm. In [11, 12] a penalty-based model is introduced for unfinished workload and competitive ratios were derived for various algorithms.

All of the works mentioned above require continuous processing of packets, i.e., each packet can be processed, paused, and restarted at any point in time *preemptively*. In [13], a slotted single queue system is considered in which all packets have unit length and uniform weights, and it is shown that EDF has the same performance as the optimal off-line algorithm. In [14, 15], the same discrete model is considered with packets having heterogenous weights and it is shown that the achievable competitive ratio is within $[0.5, \frac{\sqrt{5}-1}{2}]$. Furthermore, it is shown that the lower bound 0.5 is achieved by the largest weight first policy and a lex-optimal scheduling policy is provided.

There have also been a few works that have investigated the problem of scheduling packets with deadlines in the multihop setting. In [16], the authors investigate the problem of online scheduling of sessions with known injection rates, given deadlines and fixed routes. They first give a necessary condition for feasibility of sessions and then propose an algorithm under which most sessions are scheduled without violating the deadline with high probability when the necessary condition for feasibility is satisfied. The algorithm they proposed is called coordinated EDF. The main idea of coordinated EDF is to assign virtual deadline within certain region (that is related to the session parameters such as injection rate, the actual arrival time and deadline) for the first hop of each packet uniformly at random, and then assign virtual deadlines of the remaining hops based on the virtual deadline of first hop regulated by constants that are related to the session parameters. Upon packet scheduling for each link, the packet with earliest virtual deadline of the hop index at that link is transmitted. In [17], it has been shown that a modified version of EDF achieves the same performance as the optimal off-line algorithm for any arrival sample path over an uplink tree with uniform link capacities and packet weights.

In the first part of our paper, we consider under-loaded arrivals but allow links to have *heterogenous link capacities* and show that EDF has the same performance as that of the optimal off-line algorithm. In the second part, we consider a general multihop network, where simple heuristics such as EDF, minimum number of remaining hops first, minimum remaining time till expiration first, or Largest Weight First do not have provable efficiency. Achieving provable efficiency in a general network topology requires a joint consideration of the factors such as deadlines, packet weights, and path lengths etc. Our approach to the problem with the general multihop topology is motivated by competitive routing in virtual circuit network [18]. In the competitive routing model, link bandwidth is the resource to be allocated. By viewing the time slots as resources, the packet scheduling problem can be transformed into a resource allocation problem. Our approach has a couple of fundamental differences with the one in [16]. Firstly, our scheme is not based on the prior knowledge of the packet injection rates, as is the case in [16]. Secondly, we investigate both under-loaded and over-loaded arrivals, whereas in [16], the authors study under-loaded arrivals.

To summarize our main contributions in this paper:

- We show that EDF has the same performance of the optimal off-line algorithm under an uplink tree with heterogenous link capacities for any under-loaded arrivals.
- We develop a competitive ratio based admission control and packet scheduling framework that has low complexity and is $O(P_M \log P_M)$-competitive[2] where P_M is the maximum route length among all packets, under general multihop network topologies and arrival samples. Moreover, we show that no online algorithm can achieve a performance scaling better than $O(P_M \log P_M)$.

II. Problem Statement

We study the packet scheduling problem with hard deadlines in a general multihop network topology represented by a directed graph, as shown in Fig. 1. We assume a time slotted system. The arrival sample path consists of K packets, where each packet $i \in \{1, 2, \ldots, K\}$ (the packet set is indexed in the order of arrival times of the packets) is associated with four parameters (a_i, d_i, ρ_i, P_i). Here, a_i and d_i are the arrival time and the deadline, respectively, both of which are given in slot indices (important notations are summarized in Table I). We allow each packet i to have a weight ρ_i, which is an arbitrary real number that represents the importance of the packet. We assume that each packet i is routed through a predetermined path[3] P_i from its source node to its destination, where P_i denotes the set of links through which the packet traverses in order. We assume infinite packet buffers at all nodes. If packet i is still at a non-destination node by the end of slot d_i, then its timer expires and it is deleted from the network. Note that the arrival sample consists of finite number of packets, when all packets have finite deadlines, the packet queues in the network will always remain bounded. We assume that each packet has an identical (unit) length and each link in the network can serve an integer number (possibly different for different links) of packets at a given time slot.

We let p denote a service policy that specifies the packets to be transmitted over each link and the packets to be admitted to the system. We then define the indicator function for any policy p, to identify whether a packet reaches its destination

[2] $O(x)$-competitive means the competitive ratio goes to 0 at least as fast as $x \to \infty$.

[3] Our framework can be generalized when there are multiple candidate routes to choose.

TABLE I
LIST OF SYMBOLS

K	number of packets
a_i	arrival time of packet i
d_i	deadline of packet i
ρ_i	weight of packet i
P_i	routing path of packet i
1_i^p	indicator of packet i being successfully received under policy p
R^p	revenue of policy p
r^p	competitive ratio of policy p
s_i	average slack time per hop for packet i
$I_l(i,j)$	indicator of packet j using unit resource of packet i
$c_l(i,j)$	cost of packet j using unit resource of packet i
S_i	maximum possible per hop delay of packet i
$\tilde{I}_l(i,j)$	indicator of packet j may use unit resource of packet i
$h_l(i)$	hop index of link l in the route of packet i
$t_l(i)$	reserved time slot fore transmission at link l for packet i

within its deadline as:

$$1_i^p = \begin{cases} 1 & \text{if } i \text{ reaches its destination before the end of } d_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the weighted revenue gained by the successfully received packets as:

$$R^p = \sum_{i \in \{1,2,\ldots,K\}} \rho_i 1_i^p. \tag{2}$$

Let $\mathcal{P}_{\text{online}}$ be the set of online policies. Our **objective** is to solve

$$\max_{p \in \mathcal{P}_{\text{online}}} R^p. \tag{3}$$

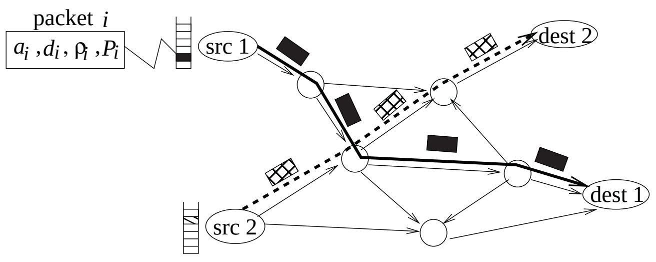


Fig. 1. General Network Topology with Multiple Source-Destination Pairs

III. MOTIVATING EXAMPLE

An *optimal off-line algorithm* is one that has the entire arrival sample path available non-causally and finds a schedule with maximum revenue among all algorithms. The optimal off-line algorithm is a conceptual tool, which is typically used as a measuring standard against which the performance of online algorithms can be compared. This is typically used in competitive ratio literature. The competitive ratio r^p of an on-line algorithm p is defined as the minimum ratio of the achieved revenue for the on-line algorithm to the revenue of the optimal off-line algorithm, where the minimization is over all possible arrival patterns:

$$r^p = \min_{v \in \mathcal{V}} \frac{R^p(v)}{R^{\text{offline}}(v)}$$

Here, $R^p(v)$ and $R^{\text{offline}}(v)$ are revenue of online algorithm p and optimal offline algorithm under arrival sample v, respectively. $\mathcal{V}$ denotes the set of all possible arrival samples.

In this section, we provide an example that shows that (1) on-line scheduling with deadlines is a difficult problem, and (2) even for very simple topologies, there may exist no on-line policy that achieves the performance of the optimal off-line algorithm (i.e., we show that the competitive ratio < 1). This is even valid for feasible arrival patterns (i.e., arrival patterns such that all packets can be served by their deadlines by the optimal offline algorithm), as we will illustrate in this example.

Example 1: Consider a line network $1 \rightarrow 2 \rightarrow 3$ and suppose that the link capacity of each link is 1, as shown in Fig. 2. Initially at node 1, there are two packets k_1 and k_2 with deadline $d_1 = 2, d_2 = 4$ whose destinations are node 2 and 3, respectively. Suppose that node 1 transmits k_1 to node 2 in time slot 1, and that there is no arrival by the end of slot 1, then node 1 transmits k_2 to node 2 in slot 2. Let an "adversary" inject a packet k_3 at node 2 by the end of slot 2 with deadline $d_3 = 3$ whose destination is node 3. Then, node 2 transmits k_3 to node 3 in slot 3. Further let the adversary inject a packet k_4 at node 2 by the end of slot 3 with deadline $d_4 = 4$ whose destination is node 3. Then, by the end of slot 4, either k_2 or k_4 expires. However, this arrival sample is feasible since the off-line algorithm transmits k_2 in slot 1 and all four packets are able to reach their destinations within their deadlines. Similarly if node 1 transmits k_2 to node 2 in time slot 1. Let the adversary inject a packet k_3 at node 1 by the end of slot 1 with deadline $d_3 = 2$ (whose destination is node 2). Then, by the end of slot 2, either k_1 or k_3 expires. However, this arrival sample is also feasible since the off-line algorithm transmits k_1 in slot 1 and all three packets are able to reach their destinations before their deadlines. This means that under this scenario, no matter what online decision node 1 makes in slot 1, the adversary can always chooses future arrivals so that the online decision is worse than the optimal off-line algorithm even though the entire arrival sample is feasible.

One of the main conclusion one can draw from this example is that, there may exist no on-line algorithm that achieves the same performance as an optimal off-line algorithm even for simple network settings. This motivates our study for developing online algorithms that have a provable (non-zero) competitive ratio, relative to the optimal off-line algorithm for a general network topology and arrival patterns. In the rest of this paper, we begin with a scenario under which there exists an online algorithm that achieves the same performance as the optimal off-line algorithm. We provide such an algorithm. We then develop an online algorithm that is provable efficient for a general multihop network.

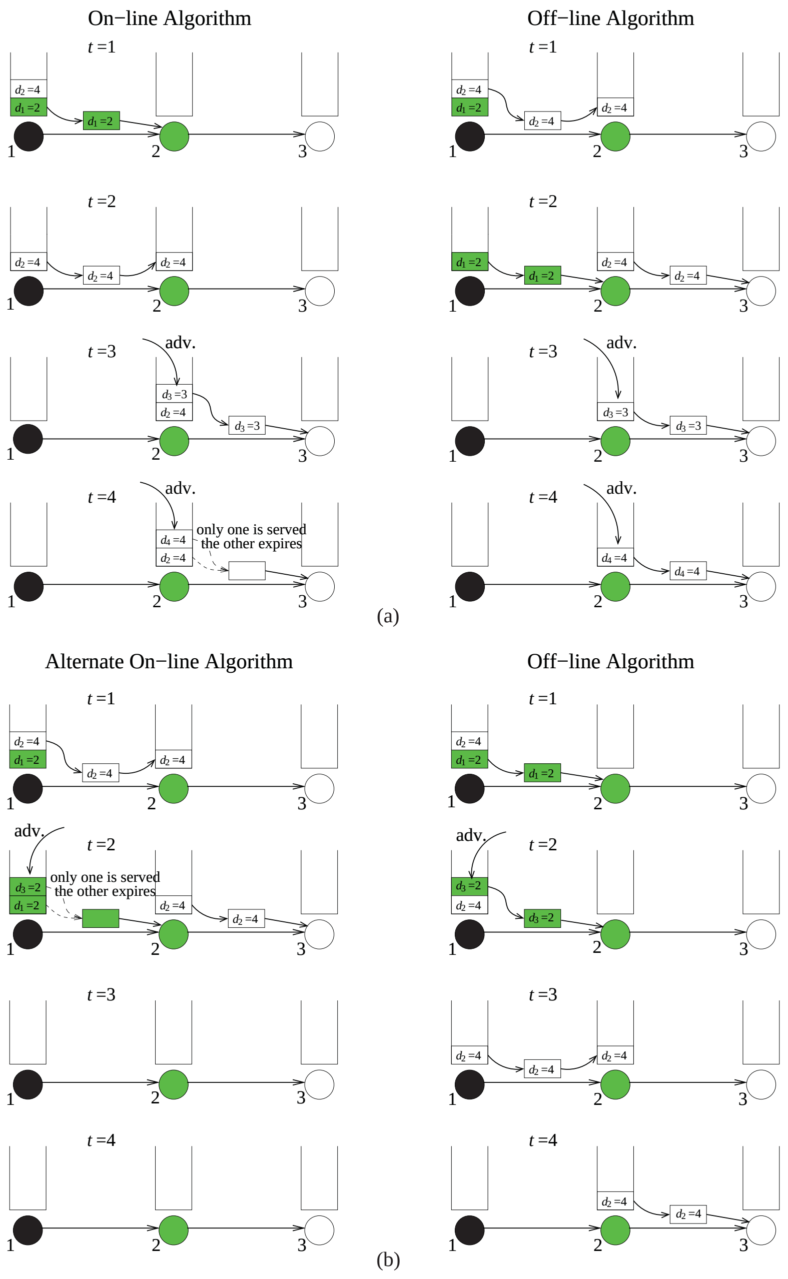


Fig. 2. An example of packet scheduling with deadlines in a line network. Four time slots are considered for two alternate sample paths. The deadline is marked in the packets and the destinations are marked with matching coloring between the packets and the nodes. The two sample paths illustrate that, no matter how an on-line algorithm schedules packets, there exists an arrival sample path that leads to a suboptimal selection with respect to the optimal off-line algorithm.

Note that, there are other topologies for which there exists no on-line algorithm with a competitive ratio identical to 1 such as:

- *Down-link tree (destinations of all packets are leaves of the tree and source nodes can be any other nodes) even with identical link capacities and under-loaded arrivals*
- *Line network with multiple flow destinations even with under-loaded arrivals*
- *Uplink tree (destination of each packet is the root of the tree and source nodes can be any other nodes) with various link capacities and overloaded arrivals.*

The details of all three of these examples can be found in Appendix C.

IV. Optimal Packet Scheduling in Uplink Tree Networks with Under-Loaded Arrivals

While it is not possible in most cases for an on-line algorithm to match the performance of the optimal off-line algorithm, in certain cases this is indeed possible. In this section, we specify such a case and find the optimal on-line algorithm with a competitive ratio of 1. In particular, we consider an uplink tree network as shown in Fig. 3. Each packet i arrives at an arbitrary non-root node in slot a_i and is destined to the root through multiple hops within its deadline d_i. For under-loaded arrivals, the packet weights are not important since an optimal algorithm serves all packets anyway. Without loss of generality, we assume that all packets have the same weight, then our objective reduces to maximizing the throughput, i.e.,

$$\max_{p\in\mathcal{P}_{\text{online}}} R^p \equiv \max_{p\in\mathcal{P}_{\text{online}}} \sum_{i\in\{1,2,\ldots,K\}} 1_i^p.$$

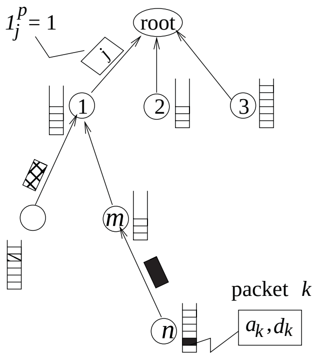


Fig. 3. An Uplink Tree

Definition 1: The *slack time* of packet i in time slot $t \in [a_i, d_i]$ is equal to $d_i - t - h_i(t) + 1$, where $h_i(t)$ is the number of hops from the node at which packet i resides in time slot t to the destination of i.

The slack time can be viewed as the extra time that the packet has above the minimum remaining time needed for a packet to reach its destination.

Definition 2: Policy p is called a *work-conserving* packet scheduling policy if it keeps each link fully utilized in each time slot, as long as the queue feeding the link contains sufficiently many unexpired packets.

Lemma 1: For any non-work-conserving policy, there exists a work conserving policy that achieves an identical or a higher throughput under any given arrival pattern.

The intuition behind Lemma 1 is fairly clear, since non-work-conserving policies unnecessarily waste resources. The detailed proof can be found in Appendix A. From Lemma 1, we only need to focus on work conserving packet scheduling policies to solve the problem, i.e., $\max_p R^p \equiv \max_{p\in\mathcal{C}} R^p$, where $\mathcal{C}$ is the set of work conserving packet scheduling policies.

Definition 3: A *work conserving earliest deadline first (WC-EDF)* policy is one in which each node transmits the largest possible set of packets that the link capacity allows with the earliest deadlines among all the packets in its packet queue.

We next state a result on WC-EDF from [17].

Theorem 1: [17] For an uplink tree with identical integer-valued link capacity, given any arrival sample path (either under-loaded or over-loaded), the WC-EDF policy that only serves packets with non-negative slack times in each slot achieves the same performance as the optimal off-line algorithm in maximizing the throughput.

This theorem, proven in [17], is for an uplink tree with identical link rates. In the following theorem, we extend the result to the scenario where links may have different rates (as long as they are integer number of packets/time slot) for under-loaded traffic.

Theorem 2: For an uplink tree with possibly different link capacities that are integer number of packets per time slot, for all under-loaded arrivals, the WC-EDF policy ensures that all packets reach their destinations before their deadlines.

The detailed proof of Theorem 2 can be found in Appendix B. Theorem 1 holds for identical link capacities and its proof is based on induction on links. Since Theorem 1 does not hold for heterogeneous link rates, the proof techniques for Theorem 2 are fairly different from those in Theorem 1. Theorem 2 is proven by a careful classification and analysis of all possible arrival sample paths.

Theorem 2 shows that under WC-EDF, all packets for any under-loaded arrival sample path reach their destinations before their deadlines and hence generates the same throughput as the optimal off-line algorithm. Note that a biproduct of Theorem 2 is that it can be used to test whether an arrival sample is feasible.

V. Competitive Packet Scheduling for General Topologies and Arrival Patterns

In this section, we focus on a general network topology (e.g., that shown in Fig. 1). We will propose a scheduling strategy, evaluate its performance, and show that the performance of our strategy is order optimal, i.e., no on-line algorithm achieves a better scaling law for the competitive ratio, in terms of the maximum route length.

For any link $l \in P_i$, let $h_l(i)$ denote the hop index in the route of packet i. For ease of notation, in this section, we present the main results for the case in which all the links have unit capacity[4]. We allow packets to have different weights $\rho_i,\ i = 1, 2, \ldots, K$, and our objective function is as stated in Equation (3).

Upon the arrival time of each packet into the network, the controller of its source node decides whether to accept or reject this packet. If there are multiple packets arriving at the network in the same time slot, we assume the controllers of different packets make decisions at different instances (in the same time slot) in the same order as the packet index. If packet i is accepted, then each link $l \in P_i$ needs to reserve a time slot so that packet i will be transmitted through link l in this reserved slot. The reserved time slot is not changed in the subsequent time slots. Let $t_i(l)$ denote the index of this reserved time slot in which packet i is transmitted through link $l \in P_i$ if i is

[4]Our results can be generalized when link rates C_l are distinct, as long as there are an integer number of packets per time slot by replacing s_i with $C_l s_i$ in Equation (6), (7), (8), and in the definition of *Cost*$_j$ in Algorithm 1.

accepted. Define for any i, j, l the following indicator

$$I_l(i,j) = \begin{cases} 1 & \text{if packets } i,j \text{ are accepted; } l \in P_i, l \in P_j; \\ & t_j(l) \in [a_i + (h_l(i)-1)s_i, a_i + h_l(i)s_i - 1] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where

$$s_i = \left\lfloor \frac{d_i - a_i + 1}{|P_i|} \right\rfloor \tag{5}$$

is the average slack time per hop for packet i with $|P_i|$ as the number of total hops on its route. Note that $d_i - a_i + 1$ is the maximum allowable end to end delay for packet i in the network. If we divide this delay evenly on each hop, then s_i is the maximum allowable delay on each hop and $[a_i + (h_l(i) - 1)s_i, a_i + h_l(i)s_i - 1]$ is the set of time slots that i can use to be transmitted through link l. From another perspective, a time slot can be viewed as a resource and $[a_i + (h_l(i) - 1)s_i, a_i + h_l(i)s_i - 1]$ is then the set of available resources for packet i and s_i is the total amount of resources at each link on the route of i. When both i and j are accepted and link l is in the route of both i and j, if the reserved transmission slot of j takes one resource in i's resource set at link l, then the indicator $I_l(i, j)$ becomes 1. This means packet j consumes one unit of resource of packet i at link l.

We define the *cost* (the exponential cost metric is motivated by [18]) of packet j taking a resource of i at link l as

$$c_l(i,j) = s_i(\mu^{\lambda_l(i,j)} - 1), \tag{6}$$

where μ is a control parameter (we will discuss how to choose the value of μ, later in this section) and

$$\lambda_l(i,j) = \sum_{m<j} \frac{I_l(i,m)}{s_i} \tag{7}$$

is the fraction of packet i's resources that have already been taken before arrival of packet j at link l. By letting $\lambda_l(i, 1) = 0$ for all i, l, we have the following recursive relationship:

$$\lambda_l(i,j+1) = \lambda_l(i,j) + \frac{I_l(i,j)}{s_i}, \tag{8}$$

i.e., $\lambda_l(i, j)$ and thus $c_l(i, j)$ is increasing in j for any given i and l. This is to be expected, since the packet arrival sequence is indexed in the order of arrival times of packets, so a later j should result in giving more time for packet i to consume of of its resources.

Before describing our algorithm, we further need to define for $i \neq j$ that

$$\tilde{I}_l(i,j) = \begin{cases} 1 & \text{the intersection of the intervals} \\ & \left[a_j + h_l(j) - 1, d_j - |P_j| + h_l(j)\right] \text{ and} \\ & \left[a_i + (h_l(i)-1)s_i, a_i + h_l(i)s_i - 1\right] \\ & \text{are nonempty; } l \in P_i, l \in P_j; \ i \text{ is accepted} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and

$$\tilde{I}_l(j,j) = 1, \ \forall j, \ \ l \in P_j. \tag{10}$$

Note that $\left[a_i + h_l(i) - 1, d_i - |P_i| + h_l(i)\right]$ is the range of time slots that packet i can possibly remain enqueued in the queues of link l for it to reach the destination before its deadline under any algorithm. Define

$$S_i \triangleq d_i - a_i - |P_i| + 2 \tag{11}$$

to be the maximum possible delay of packet i at each link $l \in P_i$ and it is easy to see that s_i, the amount of resources at each link on the route of i is less than S_i, i.e., $s_i \le S_i$ for all i. Variable $\tilde{I}_l(i, j)$ indicates whether packet j may take a resource of packet i under any possible scenario. Note that $t_l(j) \in \left[a_j + h_l(j) - 1, d_j - |P_j| + h_l(j)\right]$ for all $l \in P_j$ for any scheduler, since allocating any time slot out of this interval to transmit j over $l \in P_j$ will lead to the expiration of j. Hence, one can see that $I_l(i, j) \le \tilde{I}_l(i, j)$ for all i, j, l from Eqs. (4), (9), and (10).

Algorithm 1 Admission Control and Packet Scheduling

```
Upon arrival time of packet $j$, let $\mathit{Cost}_j := \sum_l \sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)$:
1) If $\mathit{Cost}_j > \rho_j$, then reject $j$;
2) If $\mathit{Cost}_j \le \rho_j$, then accept $j$ and let $t_l(j)$ be the empty time slot with the largest index in $[a_j+(h_l(j)-1)s_j, a_j+h_l(j)s_j-1]$. Put packet $j$ into $t_l(j),\ \forall l \in P_j$;
3) Any accepted packet $j$ is transmitted through link $l \in P_j$ at time slot $t_l(j)$.
```

Our admission control and packet scheduling algorithm is described in Algorithm 1. Note that from Equation (9), we have $\sum_l \sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j) = \sum_{l\in P_j} \sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)$, i.e., the calculation only needs information on the route of packet j. For each $l \in P_j$, the calculation of the term $\sum_{i\le j} \frac{\tilde{I}_l(i,j)}{s_i} c_l(i,j)$ only requires the information of packets that may route through link l. It is also easy to see that the calculation of the cost term does not require future information for times after the arrival time of each packet j, i.e., Algorithm 1 is an online algorithm that uses only causal information. Furthermore, $\lambda_l(i, j + 1),\ i \le j$ is calculated from $\lambda_l(i, j)$ using Equation (8) when packet j is processed by Algorithm 1, and Equation (7) is only used to calculate $\lambda_l(j, j)$ upon j's arrival. The transmission slot at each hop is determined when the packet is admitted.

The basic intuition behind our algorithm is simple: We rst allocate the end-to-end delay of each packet evenly over the links along its path. The algorithm then schedules the transmission for an accepted packet in a slot within its allocated time region at each link. Consequently, the end-to-end deadline constraint is met. With this approach, the natural questions are: (1) When a packet j is accepted, is there always an empty (non-reserved) slot in $[a_j + (h_l(j) - 1)s_j, a_j + h_l(j)s_j - 1],\ \forall l \in P_j$ so that we can reserve a slot $t_l(j)$ for j at link l? (2) What is the performance of Algorithm 1 compared to the optimal off-line algorithm? We answer these questions in the following theorem. We denote the competitive ratio of our algorithm with r, i.e., $R \ge rR^*$, where R is the worst-case (over all sample paths) weighted revenue achieved by Algorithm 1 and R^* is the weighted revenue achieved by the optimal off-line algorithm.

Theorem 3: If the system parameters satisfy $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$, where $P_M = \max_i |P_i|$ is the maximum route length, $s_m = \min_i s_i$ is the minimum average slack time, $s_M = \max_i s_i$ is the maximum average slack time, $\rho_m = \min_i \rho_i$ is the minimum weight and $\rho_M = \max_i \rho_i$ is the maximum weight among all packets, then every packet accepted by Algorithm 1 reaches its destination before its deadline. Furthermore, Algorithm 1 achieves competitive ratio $r = \Big[2\big(2(s_M P_M+1)+1\big)\big(1+\frac{\rho_M}{\rho_m}s_M\big)\log\big(2\frac{\rho_M}{\rho_m}s_M P_M+1+\epsilon\big)+1\Big]^{-1}$, where $\epsilon > 0$ is an arbitrarily small constant.

Hence, our algorithm is $O(P_M \log P_M)$-competitive when the condition of Theorem 3 is satis ed, where P_M is the maximum route length. The assumption $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ in Theorem 3 imposes an upper bound on the maximum route length P_M for the validity of the provided competitive ratio. According to the condition, P_M is upper bounded by an exponential function of the minimum average slack time s_m. Despite the fact that the competitive ratio is valid only when the condition is met, in Section VI, we will illustrate using numerical examples that our algorithm achieves a high performance relative to the optimal off-line algorithm, even when this condition is not satis ed.

To prove this theorem, we rst make the following transformation. With condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$, we can choose μ so that $\log(\mu) \le s_m$ and $\mu > 2\frac{\rho_M}{\rho_m}s_M P_M + 1 \ge 1$, i.e., $\frac{\mu-1}{2P_M} > \frac{\rho_M}{\rho_m}s_M \ge s_M$. Then, we choose a factor, $F \in \left[\frac{2P_M s_M}{\rho_m}, \frac{\mu-1}{\rho_M}\right)$, and normalize the weights ρ_i for all i with factor F and use $F\rho_i$, instead of ρ_i for all i in the problem (the objective is still equivalent to the original one). With this change, we have $\log(\mu) \le s_m \le s_M \le \frac{\rho_m}{2P_M} \le \frac{\rho_M}{2P_M} < \frac{\mu-1}{2P_M}$, i.e.,

$$I_l(i,j) \le 1 \le \frac{s_m}{\log(\mu)} \le \frac{s_i}{\log(\mu)}, \text{ for all } i,j; \tag{12}$$

$$2|P_j|s_M \le 2P_M s_M \le \rho_m \le \rho_j, \text{ for all } j; \tag{13}$$

$$\rho_j \le \rho_M < \mu - 1, \text{ for all } j. \tag{14}$$

We now provide the following sequence of lemmas to prove Theorem 3. First, we prove that, if a packet is admitted by Algorithm 1, than it is successfully served within its deadline:

Lemma 2: Under the same condition of Theorem 3, if j is accepted by Algorithm 1, then there exists at least one time slot in the interval $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1]$ that has not been reserved by other accepted packets for each $l \in P_j$.

Proof: Suppose j is the rst packet that is accepted by Algorithm 1 but there exists $l \in P_j$ so that all time slots in the interval $[a_j + (h_l(j)-1)s_j, a_j + h_l(j)s_j - 1]$ are occupied by other accepted packets when j is accepted. From Eqs. (4) and (7), we have $\lambda_l(j,j) = 1$ and $\frac{c_l(j,j)}{s_j} = \mu^{\lambda_l(j,j)} - 1 \ge \mu - 1$. Note that $\tilde{I}_l(j,j) = 1$ by Equation (10), combined with Equation (14), we then have

$$\sum_{l'}\sum_{i' \le j}\frac{\tilde{I}_{l'}(i',j)}{s_{i'}}c_{l'}(i',j)$$
$$\ge \frac{\tilde{I}_l(j,j)}{s_j}c_l(j,j) = \frac{c_l(j,j)}{s_j} \ge \mu - 1 > \rho_j,$$

i.e., j is rejected by Algorithm 1 which is a contradiction. ■

Then, we nd an upper bound on the difference between the weighted revenues achieved by the optimal off-line algorithm and by our algorithm:

Lemma 3: Let $\mathcal{Q}$ denote the set of packets that are rejected by Algorithm 1 but successfully received by the optimal off-line algorithm, then $\sum_{j\in\mathcal{Q}}\rho_j \le \left(2\frac{S_M}{s_m}+1\right)\sum_l\sum_i c_l(i,K)$ under the same condition of Theorem 3, where K is the last packet of the arrival sample.

Proof: For any $j \in \mathcal{Q}$, from Algorithm 1 and the fact that $c_l(i,j)$ is increasing in j, given any i and l, i.e., $c_l(i,j) \le c_l(i,k),\ \forall l,i$ if $j \le k$, we have

$$\rho_j < \sum_l\sum_{i\le j}\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,j) \le \sum_l\sum_i\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,j)$$
$$\le \sum_l\sum_i\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,K).$$

Consider any packet i and any link $l \in P_i$, if i is rejected by Algorithm 1, then $\tilde{I}_l(i,j) = 0,\ \forall j \ne i$ and $\tilde{I}_l(i,i) = 1$, we then have $\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i} \le \frac{1}{s_i}$; otherwise, we have $\tilde{I}(i,j) = 1$ only for j with $a_i + (h_l(i)-1)s_i \le d_j - |P_j| + h_l(j)$ and $a_j + h_l(j) - 1 \le a_i + h_l(i)s_i - 1$. Let $t_l^*(j)$ be the time slot in which packet j is transmitted through link $l \in P_j$ under the optimal off-line algorithm. Note that $t_l^*(j) \in [a_j + h_l(j) - 1, d_j - |P_j| + h_l(j)]$ since this interval is the maximum allowable transmission interval for the successful reception of j under all algorithms. Furthermore, if $t_l^*(j) < a_i + (h_l(i)-1)s_i - S_j + 1$ or $t_l^*(j) > a_i + h_l(i)s_i - 1 + S_j - 1$, then it means $d_j - |P_j| + h_l(j) \le t_l^*(j) + S_j - 1 < a_i + (h_l(i)-1)s_i$ or $a_j + h_l(j) - 1 \ge t_l^*(j) - S_j + 1 > a_i + h_l(i)s_i - 1$, i.e., $\tilde{I}(i,j) = 0$. Therefore, we must have $t_l^*(j) \in [a_i + (h_l(i)-1)s_i - S_j + 1, a_i + h_l(i)s_i - 1 + S_j - 1]$ if $\tilde{I}(i,j) = 1$, then $\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i} \le \frac{2S_j+s_i-2}{s_i}$. By summing over all $j \in \mathcal{Q}$ and combining with the fact $\tilde{I}_l(i,j) = 0,\ \forall l \notin P_i$, we have

$$\sum_{j\in\mathcal{Q}}\rho_j < \sum_l\sum_i c_l(i,K)\sum_{j\in\mathcal{Q}}\frac{\tilde{I}_l(i,j)}{s_i}$$
$$\le \left(2\frac{S_M}{s_m}+1\right)\sum_l\sum_i c_l(i,K),$$

where $S_M = \max_i S_i$ is the maximum link delay among all packet and $s_m = \min_i s_i$ is the minimum average slack time among all packets. ■

Finally, we derive a lower bound on the achieved weighted revenue by our algorithm:

Lemma 4: Let $\mathcal{A}$ denote the set of packets that are accepted by Algorithm 1, then $\sum_l\sum_i c_l(i,K) \le$

$2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\sum_{j\in\mathcal{A}}\rho_j$ under the same condition of Theorem 3.

Proof: Note that for any l, i and $j \in \mathcal{A}$, we have

$$\begin{aligned} c_l(i,j+1)-c_l(i,j) &= s_i\mu^{\lambda_l(i,j)}\left(\mu^{\frac{I_l(i,j)}{s_i}}-1\right) \\ =&s_i\mu^{\lambda_l(i,j)}\left(2^{\log(\mu)\frac{I_l(i,j)}{s_i}}-1\right) \le \mu^{\lambda_l(i,j)}I_l(i,j)\log(\mu) \\ =&\left(\frac{c_l(i,j)}{s_i}+1\right)I_l(i,j)\log(\mu), \end{aligned} \tag{15}$$

where the inequality is from Equation (12) and the fact $2^x-1 \le x,\ x\in[0,1]$. The last equality is from Equation (6).

Recall that $I_l(i,j) \le \tilde{I}_l(i,j),\ \forall i,j,l$ by Equation (4) and Equation (9); $c_l(i,j) \le c_l(i,i),\ \forall i > j$ by Equation (6), Equation (8) and Equation (4); and $\sum_l\sum_{i\le j}\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,j) \le \rho_j,\ \forall j \in \mathcal{A}$ from Algorithm 1. From Equation (4), we have $\sum_i I_l(i,j) = \sum_{i\in\mathcal{A}} I_l(i,j),\ \forall j\in\mathcal{A}$. From Lemma 2, $t_l(i) \in [a_i+(h_l(i)-1)s_i, a_i+h_l(i)s_i-1]$ for any $i\in\mathcal{A}$. Note that in order to have $I_l(i,j) = 1$, we must have $a_i+(h_l(i)-1)s_i \le t_l(j) \le a_i+h_l(i)s_i-1$. For any $i\in\mathcal{A}$, if $t_l(i) < t_l(j)-s_i+1$ or $t_l(i) > t_l(j)+s_i-1$, then it means $a_i+h_l(i)s_i-1 \le t_l(i)+s_i-1 < t_l(j)$ or $a_i+(h_l(i)-1)s_i \ge t_l(i)-s_i+1 > t_l(j)$, i.e., $I_l(i,j)=0$. Therefore, if $I_l(i,j) = 1,\ \forall i\in\mathcal{A}$, then $t_l(i)\in[t_l(j)-s_i+1, t_l(j)+s_i-1]$, i.e., $\sum_l\sum_i I_l(i,j) = \sum_l\sum_{i\in\mathcal{A}} I_l(i,j) \le 2|P_j|s_i \le 2|P_j|s_M \le \rho_j$ using Equation (13), and $\sum_{i\in\mathcal{A}}\frac{\rho_i}{\rho_j}I_l(i,j) \le 2\frac{\rho_M}{\rho_m}s_M$. For any l, i and $j\in\mathcal{A}$, by summing over l and i of Equation (15), we have

$$\begin{aligned} &\sum_l\sum_i\Big[c_l(i,j+1)-c_l(i,j)\Big] \\ \le&\log(\mu)\left[\sum_l\sum_i\frac{I_l(i,j)}{s_i}c_l(i,j)+\sum_l\sum_i I_l(i,j)\right] \\ \le&\log(\mu)\left[\sum_l\sum_{i\le j}\frac{\tilde{I}_l(i,j)}{s_i}c_l(i,j)+\right. \\ &\left.\sum_l\sum_{i>j}\frac{I_l(i,j)}{s_i}c_l(i,j)+2|P_j|s_M\right] \\ \le&\log(\mu)\left[\rho_j+\sum_{\substack{i>j,i\in\mathcal{A}\\ I_l(i,j)=1}}\sum_l\frac{\tilde{I}_l(i,i)}{s_i}c_l(i,i)+2|P_j|s_M\right] \\ \le&\log(\mu)\left[\rho_j+\sum_{\substack{i>j,i\in\mathcal{A}\\ I_l(i,j)=1}}\sum_l\sum_{k\le i}\frac{\tilde{I}_l(k,i)}{s_k}c_l(k,i)+2|P_j|s_M\right] \\ \le&\log(\mu)\left[\rho_j+\rho_j\sum_{i\in\mathcal{A}}\frac{\rho_i}{\rho_j}I_l(i,j)+2|P_j|s_M\right] \\ \le&\log(\mu)\left[\rho_j+2s_M\frac{\rho_M}{\rho_m}\rho_j+2|P_j|s_M\right] \\ \le&2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\rho_j, \end{aligned}$$

and combined with $c_l(i,j+1)-c_l(i,j)=0,\ \forall j\notin\mathcal{A}$, we have

$$\begin{aligned} \sum_l\sum_i c_l(i,K) =& \sum_l\sum_i\sum_j\Big[c_l(i,j+1)-c_l(i,j)\Big] \\ =&\sum_l\sum_i\sum_{j\in\mathcal{A}}\Big[c_l(i,j+1)-c_l(i,j)\Big] \\ \le&2\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)\sum_{j\in\mathcal{A}}\rho_j. \quad ■ \end{aligned}$$

Proof of Theorem 3: From Lemma 2, we see that for every accepted packet j, there exists an unreserved time slot $t_l(j)$ in the interval $[a_j+(h_l(j)-1)s_j, a_i+h_l(j)s_j-1]$ for transmission at any $l\in P_j$. With this allocation of the total slack time on each hop, it is apparent that j can reach its destination before deadline d_j, if it is transmitted over link l in slot $t_l(j)$ for all $l\in P_j$, which is indeed the case for the admitted packets.

Lemma 2 proves the rst part of the theorem. The remaining part of the theorem is proved in two steps: Combining Lemmas 2, 3 and 4, we have

$$\begin{aligned} R^* \le& \sum_{j\in\Omega}\rho_j+\sum_{j\in\mathcal{A}}\rho_j \\ \le&\left(2\frac{S_M}{s_m}+1\right)\sum_l\sum_i c_l(i,K)+\sum_{j\in\mathcal{A}}\rho_j \\ \le&\left[2\left(2\frac{S_M}{s_m}+1\right)\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)+1\right]\sum_{j\in\mathcal{A}}\rho_j \\ =&\left[2\left(2\frac{S_M}{s_m}+1\right)\left(1+s_M\frac{\rho_M}{\rho_m}\right)\log(\mu)+1\right]R \triangleq \frac{R}{\bar{r}}. \end{aligned}$$

Note that $s_m \ge 1$ for a slotted system. From Eqs. (5) and (11), we have $S_i \le (s_i+1)|P_i|-|P_i|+1 \le s_i|P_i|+1$ and then $S_M \le s_M P_M+1$. Recall that $\mu > 2\frac{\rho_M}{\rho_m}s_M P_M+1$. By letting $\mu = 2\frac{\rho_M}{\rho_m}s_M P_M+1+\epsilon$, where $\epsilon > 0$ can be arbitrarily small, we have $\bar{r} \ge \Big[2\big(2(s_M P_M+1)+1\big)\big(1+\frac{\rho_M}{\rho_m}s_M\big)\log\big(2\frac{\rho_M}{\rho_m}s_M P_M+1+\epsilon\big)+1\Big]^{-1}$, which is identical to the competitive ratio r. Hence, our algorithm is $O(P_M\log P_M)$-competitive. ■

Theorem 4: In a general multihop network topology with general arrival samples, all online algorithms are $\Omega(P_M\log P_M)$-competitive.

Proof: Consider a line network with P_M links $l_1,\dots,l_{P_M}$ and a sequence of packets that consists of $s_M^2P_M\log\left(\frac{\rho_M}{\rho_m}s_M P_M\right)$ phases, where P_M is the maximum path length, ρ_M is the maximum packet weight, ρ_m is the minimum packet weight, and s_M is the maximum average slack time. Assume $\frac{\rho_M}{\rho_m}s_M P_M$ is a power of 2 and each link has unit capacity. Each phase $i\in[1, s_M^2P_M\log(\frac{\rho_M}{\rho_m}s_M P_M)]$ has $\lceil\frac{\rho_m}{\rho_M s_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil$ groups, each group $j\in[1,\lceil\frac{\rho_m}{\rho_M s_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil]$ has $\lceil\frac{\rho_m}{\rho_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil$ mini groups, and each mini group $k\in[1,\lceil\frac{\rho_m}{\rho_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil]$ has $\frac{s_M P_M}{\lceil\frac{\rho_m}{\rho_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil}$ packets. A packet in phase i group j mini group k has link $l_{\frac{P_M}{\lceil\frac{\rho_m}{\rho_M s_M}2^{\lfloor\frac{i}{s_M^2P_M}\rfloor}\rceil}(j-1)+1}$ as its rst hop link,

$l_{\frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil} j}$ as its last hop link, $\frac{s_M P_M}{\lceil \frac{\rho_m}{\rho_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil}$ as its end-to-end delay, and $1 + \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil}(j-1) + (k-1)$ as its arriving time slot. Then, it is easy to see that the maximum average slack time is s_M and the maximum path length is P_M. Let all packets in phase i have the same weight ρ_i. Note that weight ρ_i is obtained in phase i by using $\frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil}$ resources, then a unit weight is gained by using $\frac{1}{\rho_i} \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil}$ resources.

Let x_i be the weighted revenue obtained by the online algorithm from packets in phase i. Since there are $s_M P_M$ time slots and P_M links in total, we have $s_M P_M^2$ time resources. Therefore,

$$\sum_{i=1}^{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} x_i \frac{1}{\rho_i} \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil} \leq s_M P_M^2. \tag{16}$$

De ne $F_j = \frac{1}{\rho_j} \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{j}{s_M^2 P_M} \rfloor} \rceil} \sum_{i=1}^{j} x_i$, then

$$\begin{aligned}
&\sum_{j=1}^{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} F_j \\
=&\sum_{j=1}^{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} \frac{1}{\rho_j} \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{j}{s_M^2 P_M} \rfloor} \rceil} \sum_{i=1}^{j} x_i \\
\leq&\sum_{i=1}^{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} 2x_i \frac{1}{\rho_i} \frac{P_M}{\lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{i}{s_M^2 P_M} \rfloor} \rceil} \\
\leq&2 s_M P_M^2,
\end{aligned}$$

where the last inequality is from Equation (16). This means there exists $k \in [1, s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)]$ so that $F_k \leq \frac{2 s_M P_M^2}{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)}$ and then the weighted revenue of online algorithms

$$\begin{aligned}
\sum_{i=1}^{k} x_i =& \frac{\rho_k \lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{k}{s_M^2 P_M} \rfloor} \rceil}{P_M} F_k \text{ (by de nition of } F_k) \\
\leq& \frac{\rho_k \lceil \frac{\rho_m}{\rho_M s_M} 2^{\lfloor \frac{k}{s_M^2 P_M} \rfloor} \rceil}{P_M} \frac{2 s_M P_M^2}{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} \\
\leq& \rho_k \frac{2 s_M P_M^2}{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)}.
\end{aligned}$$

We rst let $\rho_k = \rho_m$. If $k \neq s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)$, then let $\rho_{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)} = \rho_M$. The off-line algorithm can simply serve packets in phase $s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)$ to obtain weighted revenue $\rho_M s_M P_M^2$; If $k = s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)$, then let $\rho_{s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M) - 1} = \rho_M$. The off-line algorithm can simply serve packets in phase $s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M) - 1$ to obtain weighted revenue that is larger than $\frac{1}{4} \rho_M s_M P_M^2$. Therefore, the competitive ratio is bounded by $\left[\frac{\rho_M}{8\rho_m} s_M^2 P_M \log(\frac{\rho_M}{\rho_m} s_M P_M)\right]^{-1}$ which is in the same order of r and the optimal online algorithm is $O(P_M \log P_M)$-competitive. ■

Consequently, our algorithm is order optimal in competitive ratio when the condition of Theorem 3 is satis ed.

Our competitive ratio based framework of packet scheduling with deadlines is motivated by the competitive routing model [18]. In [18], link bandwidth is the limited resource and the algorithm decides whether to admit and route packets according to the available link bandwidth along the path so that the throughput can be maximized. However, there are signi cant differences between the two models: 1) The amount of link bandwidth is xed and known at the beginning for wireline networks in the competitive routing model [18], but the time slot resources are related to the accepted packets (arrival time and deadline) and are then related to the algorithm itself (since the available time resource is related to the previously accepted packets by the algorithm); 2) A packet that arrives after the current accepted packet may get admitted and share overlapped routing hop and time slots that would be available to a potential packet arriving later. Thus, we treat a time slot as resources to be shared between packets. In an on-line algorithm, current decisions can in uence future decisions, which make it dif cult to develop such algorithms; 3) There are multiple ways of allocating the total end-to-end delay $d_j - a_j + 1$ (slack time from arrival time to deadline) of each packet j on each hop, which brings further complexities to our problem.

VI. Numerical Examples

We rst consider an uplink tree shown in Fig. 4(a) with identical link rates. There are 10000 packets, also with identical weights, and the inter-arrival times of packets are chosen to be 0 w.p. $\frac{1}{2}$ and 1 time slot w.p. $\frac{1}{2}$. We generate the initial slack time (the difference between the deadline and the arrival time, increased by 1) uniformly at random between 24 and 30. The source node of each packet is chosen uniformly at random over all non-root nodes. It is easy to see that the maximum route length $P_M = 3$ for all packets in this network. Thus, the maximum average slack time is $s_M = 30$ and the minimum average slack time is $s_m = 8$, i.e., $P_M = 3 < \frac{2^{s_m} - 1}{2 s_M \left(\frac{\rho_M}{\rho_m}\right)} = \frac{255}{60}$. As given in the proof of Theorem 3, we choose the control parameter $\log(\mu) = 7.5$ so that $\log\left(2 \frac{\rho_M}{\rho_m} s_M P_M + 1\right) = 7.4 < \log(\mu) \leq s_m = 8$. From Fig. 5(a), we can see that by increasing the normalization factor of the packets' weight, the performance of our online algorithm increases and it achieves a revenue, larger than 90% of the revenue achieved by the off-line algorithm (the performance of the optimal off-line algorithm is obtained using the algorithm in Theorem 1). Since the arrival sample is close to under-loaded in this example, admission control is unnecessary for most packets. By tuning the normalized weight so that it is large enough for the admission controller to admit almost all packets, better performance of the online algorithm can be obtained.

Similarly, for another choice of slack times, generated uniformly at random between 0 and 5, the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is violated. Yet, our online algorithm achieves a revenue, approximately 80% of that of the optimal off-line algorithm.

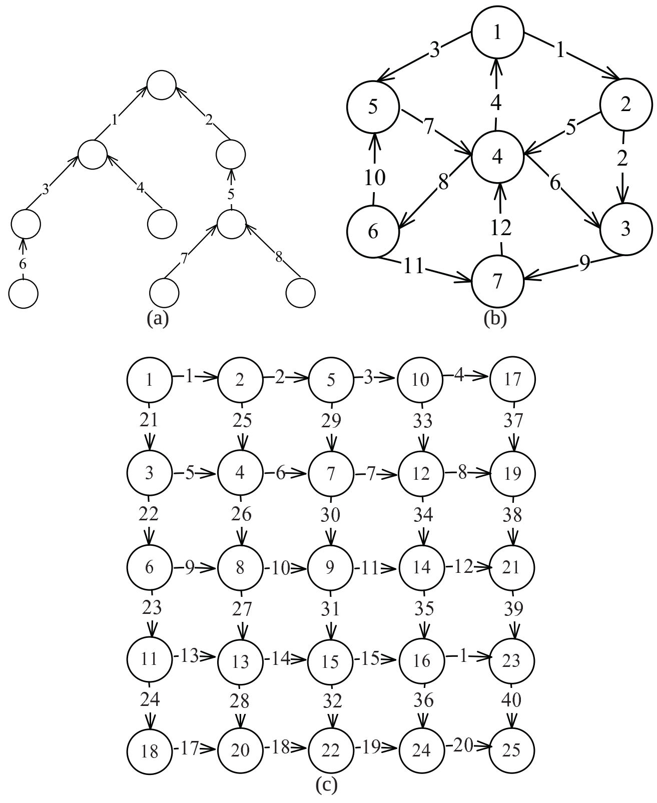


Fig. 4. Example topologies: (a) An Uplink Wired Tree, and (b) A General Wired Mutlihop Network with Multiple Source-Destination Flows (c) A Wired Grid Network

Next, we consider the (general) topology shown in Fig. 4(b). There are 10000 packets and the inter-arrival times of packets are chosen to be 0 w.p. $\frac{1}{2}$ and 1 time slot w.p. $\frac{1}{2}$. The packet weights are generated uniformly at random between 1 and 100 with $\frac{\rho_M}{\rho_m} = 100$. The given route P_i of each packet i is generated in the setup stage as follows: the source node is chosen uniformly at random among all nodes, the outgoing link of each hop is chosen uniformly at random among all possible outgoing links, and at each hop, there is $\frac{1}{2}$ probability to continue to next hop until reach the maximum route length 3. We rst generate the initial slack time uniformly at random between 45 and 50 so that the condition $P_M = 3 < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)} = \frac{2^{15}-1}{10000}$ is satis ed. Note that the region of the control parameter μ satis es $\log\left(2\frac{\rho_M}{\rho_m}s_M P_M + 1\right) = 11.55 < \log(\mu) \le s_m = 15$ in this example. With normalized packet weights (normalization factor is 300), we simulate the generated revenue as a function of $\log(\mu)$. We see from Fig. 5 (b) that our online algorithm achieves a revenue that is the same as the upper bound of the optimal off-line algorithm (we compare our scheme with the revenue upper bound, i.e., the total normalized revenue, rather than the actual revenue of the off-line algorithm due to the extremely high complexity of the calculation of the off-line algorithm). Moreover, we can see that for values of $\log(\mu) \in [0, 11.55)$ which is out of the region $(11.55, 15]$, the online algorithm still achieves the same performance of the optimal of ine algorithm. The reason is that since this arrival sample is feasible (our online algorithm can serve all arrivals successfully as shown in Fig. 5(b)), the admission controller should admit all arrivals which is exactly the special case when $\log(\mu) = 0$ and the cost term of our algorithm is always zero.

Similarly, for another choice of slack times, generated uniformly at random between 0 and 5, the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is violated. Yet, our online algorithm achieves a revenue, larger than 80% of the upper bound with appropriately chosen $\log(\mu)$.

We now modi ed the inter-arrival distribution so that it is 0 w.p. 0.8 and 1 time slot w.p. 0.2. For this more bursty arrival sample, we will see that the admission controller starts to take effect. From Fig. 5(b), it is shown that our online algorithm works best at $\log(\mu) \approx 11.55$ and it achieves around 90% of the performance upper bound. The performance decreases for either smaller or larger values of μ.

From examples (a) and (b), we can see that the assumption $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ and the region $\log\left(2\frac{\rho_M}{\rho_m}s_M P_M + 1\right) < \log(\mu) \le s_m$ of μ can be violated, the weight normalization factor is not necessarily used and the admission controller can be inactive. In practice, while the assumptions required in the proof of Theorem 3 are relaxed, the control parameters can be appropriately tuned for different scenarios so that our algorithm still has ef cient performance.

Fig. 6 compare the performance of our online algorithm, EDF, LWF and the coordinated EDF algorithm introduced in [16]. In EDF, each link transmits the packet with the earliest deadline among the packets with nonnegative slack time every time slot. In LWF, each link transmits the packet with the largest weight among the packets with nonnegative slack time every time slot. The main idea of coordinated EDF (Section 2.2 of [16]) is to assign virtual deadline within a certain region (that is related to the session parameters such as injection rate, the actual arrival time and deadline) for the rst hop of each packet uniformly at random, and then assign virtual deadlines of the remaining hops based on the virtual deadline of rst hop regulated by constants that are related to the session parameters. Upon packet scheduling for each link, the packet with earliest virtual deadline of the hop index at that link is transmitted. We use the general topology given in Fig. 4(b) for results in Fig. 6(a) and (b). We use the grid network given in Fig. 4(c) for results in Fig. 6(c) and (d). There are 10000 packets and the inter-arrival times of packets are chosen to be 0 w.p. 0.98 and 1 time slot w.p. 0.02 for results in Fig. 6(a) and (c). The inter-arrival times are exponentially distributed with

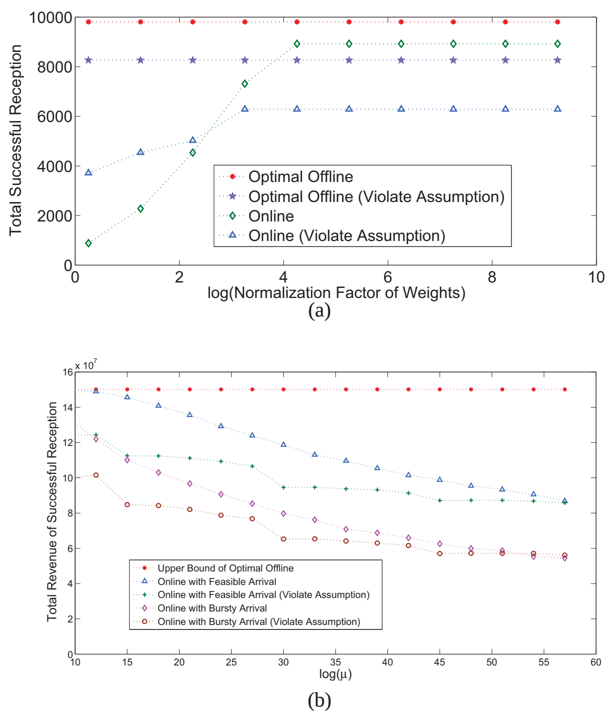


Fig. 5. Performance of Online Algorithm with (a) Different Normalized Packets Weights, and (b) Different $\log(\mu)$

rate parameter 10 for results in Fig. 6(b) and (d). The packet weights are generated uniformly at random between 1 and 2. The given route P_i of each packet i is generated as previously described for topology in Fig. 4(b). For the grid network, the given route P_i of each packet i is generated as follows: the source node is chosen uniformly at random among nodes 1-8, the outgoing link of each hop is chosen uniformly at random among all possible outgoing links, and at each hop, there is $\frac{1}{2}$ probability to continue to next hop until reach the maximum route length 5. We let all packets have the same initial slack time (end-to-end delay requirement) and we vary the slack time in our comparison. We can see that our online algorithm with carefully selected control parameters outperforms EDF, LWF and coordinated EDF in all scenarios. The revenues of all algorithms increase as the initial slack time becomes large.

Finally, we consider a line network $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ with four nodes and 3 links (identical link rates of 1 packet/time slot). The arrival sample is periodic with a period identical to 6 slots. It is illustrated in the following table:

slot	1	2	3	4	5	6
packet	$\{1,2\}$	$\{3\}$	$\{4,5,6\}$	$\{7,8\}$	$\emptyset$	$\emptyset$
$\mathbf{d_i}$	$\{6,1\}$	$\{2\}$	$\{3,5,4\}$	$\{4,5\}$	$\emptyset$	$\emptyset$
(src,dst)	$\{(1,4), (1,2)\}$	$\{(1,2)\}$	$\{(1,2), (2,4), (2,4)\}$	$\{(1,2), (2,4)\}$	$\emptyset$	$\emptyset$
$\rho_{\mathbf{i}}$	$\{100,90\}$	$\{1\}$	$\{1,200,1\}$	$\{1,50\}$	$\emptyset$	$\emptyset$

This arrival pattern repeats every 6 slots, until a total of 10000 packets arrive. We use the normalization factor $F = 12$ for the packets weights in the algorithm. It is easy to calculate the optimal weighted revenue $\frac{10000}{8} * (100+90+200+50+1+1)*12 = 6.6 \times 10^6$. We compare our online algorithm with EDF and LWF. Note that the condition $P_M < \frac{2^{s_m}-1}{2s_M\left(\frac{\rho_M}{\rho_m}\right)}$ is already violated for this arrival sample. The revenues achieved by different schemes are summarized in the following table. We chose $\log(\mu) = 10$ for our scheme:

algorithm	proposed	EDF	LWF
achieved revenue	5.9×10^6	5.3×10^6	4.4×10^6

In this case, our algorithm outperforms EDF and LWF, and it achieves about 90% of the revenue, achieved by the optimal off-line algorithm.

VII. Conclusion

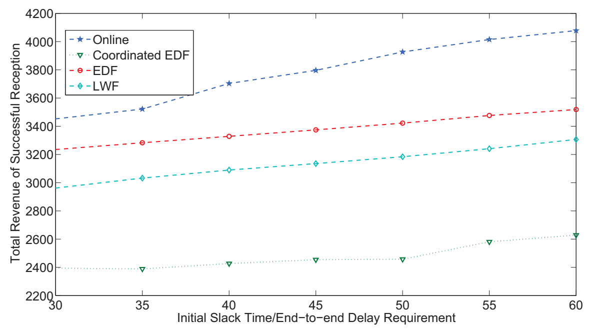
In this paper, we studied the packet scheduling problem with hard deadline constraints in multihop networks. We rst showed that, an on-line algorithm cannot achieve the performance of the off-line algorithm, even in most simple topologies. Then, we showed that this is not true for the uplink tree, and showed that WC-EDF has the same performance as the optimal off-line algorithm, given any feasible arrival sample path. We then proposed an on-line joint admission control and packet scheduling algorithm that requires only information on the route of each packet in the calculation, and has provable competitive ratio to the optimal off-line algorithm in maximizing the weighted revenue. We also show that our online algorithm achieves the highest achievable scaling law for the competitive ratio by any online algorithm. Furthermore, we show through numerical examples that our algorithm usually performs much better than the theoretical lower bound, found for the worst case arrival sample path. Our packet scheduling methodology can be applied to ow scheduling if the packet arrival of each ow is periodic and nite. Since our algorithm involves centralized coordination over the route which requires message passing and has communication overhead, the main contribution of this paper is on the theoretical aspect to investigate performance limitation of online algorithms. It is also interesting to apply ideas of this online framework to develop practical algorithms in real networking scenarios such as software de ned networks (SDN), where the control plane has centralized information and control logic.

References

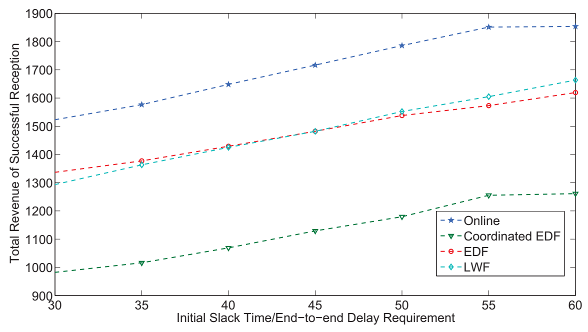
[1] C. L. Liu and J. W. Layland, "Scheduling Algorithms For Multiprogramming In a Hard-Real-Time Environment," *Journal of ACM*, vol. 20, pp. 46–61, 1973.

[2] M. Dertouzos, "Control Robotics: The Procedural Control of Physical Processes," in *IFIP Congress*, 1974, pp. 807–813.

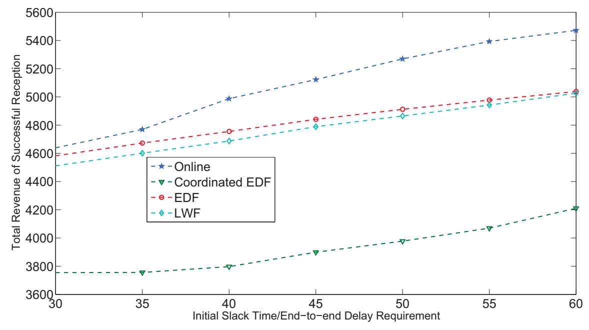
[3] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On The Competitiveness of On-Line Real-Time Task Scheduling," *Real-Time Systems*, vol. 4, pp. 125–144, 1992.



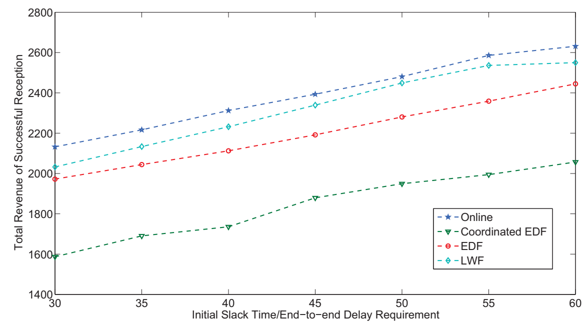
(a)



(b)



(c)



(d)

Fig. 6. Performance Comparison of Algorithms with Different Initial Slack Times under Arrivals with Bernoulli and Exponential Distributed Inter-arrival Time, respectively, in (a,b) A General Topology Network and (c,d) A Grid Network

[4] G. Koren and D. Shasha, “Dover: An Optimal On-Line Scheduling Algorithm for Overloaded Uniprocessor Real-Time Systems,” *SIAM Journal of Computing*, vol. 24, pp. 318–339, 1995.

[5] A. Bar-Noy, J. A. Garay, and A. Herzberg, “Sharing Video on Demand,” *Discrete Applied Mathematics*, vol. 129, no. 1, pp. 3–30, 2003.

[6] M. Goldwasser and B. Kerbikov, “Admission Control with Immediate Noti cation,” *Journal of Scheduling*, pp. 269–285, 2003.

[7] S. Chen, L. Tong, and T. He, “Optimal Deadline Scheduling with Commitment,” *49th Allerton Conference on Communication, Control and Computing*, September 2011.

[8] J. Ding and G. Zhang, “Online Scheduling with Hard Deadlines on Parallel Machines,” in *2nd AAIM*, 2006, pp. 32–42.

[9] J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang, “Online Scheduling of Equal-Length Jobs on Parallel Machines,” in *15th ESA*, 2007, pp. 427–438.

[10] T. Ebenlendr and J. Sgall, “A Lower Bound for Scheduling of Unit Jobs with Immediate Decision on Parallel Machines,” in *6th WAOA*, 2008, pp. 43–52.

[11] N. Thibault and C. Laforest, “Online Time Constrained Scheduling with Penalties,” in *23rd IEEE Int. Symposium on Parallel and Distributed Processing*, 2009.

[12] S. P. Y. Fung, “Online Preemptive Scheduling with Immediate Decision or Noti cation and Penalties,” in *the 16th Annual International Conference on Computing and Combinatorics, ser. COCOON10*, 2010, pp. 389–398.

[13] T. Ling and N. B. Shroff, “Scheduling Real-Time Traf c in ATM Networks,” in *Proc. of IEEE INFOCOM*, March 1996, pp. 198–205.

[14] B. Hajek, “On the Competitiveness of On-Line Scheduling of Unit-Length Packets with Hard Deadlines in Slotted Time,” in *Conference on Information Sciences and Systems*, Johns Hopkins University, March 21-23 2001, pp. 434–439.

[15] B. Hajek and P. Seri, “Lex-Optimal Online Multiclass Scheduling with Hard Deadlines,” *Mathematics of Operations Research*, vol. 30, no. 3, pp. 562–596, August 2005.

[16] M. Andrews and L. Zhang, “Packet Routing with Arbitrary End-to-End Delay Requirements,” *ACM Symposium on Theory of Computation*, pp. 557–565, 1999.

[17] P. P. Bhattacharya, L. Tassiulas, and A. Ephremides, “Optimal Scheduling with Deadline Constraints in Tree Networks,” *IEEE/ACM Transactions on Automatic Control*, vol. 42, no. 12, pp. 1703–1705, December 1997.

[18] B. Awerbuch, Y. Azar, and S. Plotkin, “Throughput Competitive Online Routing,” in *Proc. of the 34th Annual Symposium on Foundations of Computer Science*, November 1993, pp. 32–40.

Appendix A
Proof of Lemma 1

Let policy p^* be a policy that has the same performance of the optimal off-line algorithm, i.e., given any arrival sample, $R^{p^*} = \sum_{k\in\{1,2,\ldots,K\}} 1_k^{p^*} = \max_p \sum_{k\in\{1,2,\ldots,K\}} 1_k^p$.

If p^* is a work conserving packet scheduling policy, then we are done. Suppose T_0 is the rst time slot in which p^* does not schedule packets in a work conserving way. Let p be another policy that is equivalent to p^* before slot T_0, then the queue state of the network under policy p is the same as p^* at the beginning of slot T_0.

In slot T_0, there exists some node n that has a nonempty packet queue but does not schedule packets in a work conserving style under policy p^*. However, all nodes transmit packets in a work conserving way under policy p, i.e., the node n transmits more packets to its parent m than that under p^*. Let k denote a packet at node n that is scheduled under p but not scheduled under p^* in slot T_0. Then, at the beginning of slot T_0+1, k is in the descendent node of m under policy p^*.

In any slot $t > T_0$, consider any node n and let k be a packet scheduled for transmission at n under p^*, if k is in the queue of node n under policy p, then it is scheduled under p, otherwise, the packet scheduling decision under p for n is work conserving and transmits an arbitrary packet that is not scheduled under p^*. We show that any packet that arrives at the root before its deadline under p^* also arrives at the root before its deadline under p by induction. Choose $T-1 > T_0$, and assume that for any packet k, if k is at node n under p in slot $t+1$ where $t \in (T_0, T-1]$, then k is in node n or its descendent node under p^* in slot $t+1$. From the above discussion, this hypothesis holds for T_0+1 and we need to show it holds for all packets in slot $T+1$. From the way we constructed policy p, if any node n is transmitting a packet k under p^*, n is also transmitting a packet if the queue is nonempty. Furthermore, if k is in the packet queue of node n, it is transmitted under p as well. We then have the following cases:

Case 1: If k is transmitted by node n under both policies, then hypothesis holds for k in slot $T+1$;

Case 2: If k is transmitted by node n under p^* but not transmitted by p, then k is not in the packet queue of node n under p in slot T. By the hypothesis in slot T, k is in an ancestor node of n under p, then after transmission, the hypothesis still holds for k in slot $T+1$;

Case 3: If k is transmitted by p and not transmitted by p^*. By the hypothesis for k in slot T, k is in node n or its descendent node under p^*. Therefore, after k's transmission, the hypothesis still holds for k in slot $T+1$.

Therefore, for any time slot t, if any packet k is in the packet queue of node n under p in slot t, then k is in the packet queue of node n or its descendent node under p^* in slot $t+1$. This means for any packet k, if k expires under p, then k also expires under p^*, i.e., $1_k^{p^*} \le 1_k^p$ and

$$R^{p^*} = \sum_k 1_k^{p^*} \le \sum_k 1_k^p = R^p.$$

Since p^* is an optimal policy and $R^p \le R^{p^*}$, then $R^p = \sum_k 1_k^p = \sum_k 1_k^{p^*} = R^{p^*}$ which means p is also an optimal work conserving packet scheduling policy.

Appendix B
Proof of Theorem 2

Let policy p^* be a work conserving optimal off-line policy, i.e., given any under-loaded arrival sample, all the packets are successfully transmitted to the root before their deadlines under p^*. If under p^*, all nodes transmit their earliest deadline packets in all slots, then we are done. Suppose T_0 is the rst time slot in which policy p^* differs from WC-EDF p. Then, all the nodes in the network have the same set of packets in all slots before slot T_0.

We show that WC-EDF p is optimal for any under-loaded arrival sample by induction. Choose $T-1 > T_0$ and let the following holds as hypothesis $\forall t \in (T_0, T-1]$:

1) There is no packet expiring under p at the end of slot t;

2) The total number of packets at any node n under policy p^* and p are the same at the end of slot t. Any packet k at any node n under policy p can be paired with a packet k^* at the same node n under policy p^* at the end slot t;

3) For any packet pair (k_0^*, k_0) at any node n at the end of slot t, there is a sequence of packet pairs $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ at node n and its descendent nodes so that $d_{k_1^*} \le d_{k_0}, \ldots, d_{k_i^*} \le d_{k_{i-1}}, d_{k_0^*} \le d_{k_i}$ if $i \ge 1$ and $d_{k_0^*} \le d_{k_0}$ if $i = 0$.

At the beginning of slot T_0, all nodes have the same set of packets under p^* and p, and all packet pairs are common packet pairs. Consider an arbitrary node n and suppose the capacity of the link from node n to its parent node is c, i.e., the maximum number of packets that can be transmitted from node n is c in each slot. Note that the total number of packets at node n are the same under p^* and p, if this number is larger than c, then node n transmits c packets under both policies, otherwise, it transmits all packets under both policies. This means the number of transmitted packets of node n are the same under p^* and p. We use the following rule to pair the transmitted and remaining packets: If (k^*, k) forms a packet pair at the beginning of slot T_0, and k^*, k are transmitted by node n in slot T_0 under policy p^* and p, respectively, then (k^*, k) still forms a packet pair after transmission. If (k^*, k) forms a packet pair at the beginning of slot T_0, k^* is transmitted by node n in slot T_0 under p^* but k is not transmitted by p, since the total number of transmitted packets are the same for both policies, there must exist another packet pair (q^*, q) at node n so that q is transmitted by node n in slot T_0 under p but q^* is not transmitted p^*, then (k^*, q) is a transmitted packet pair by node n in slot T_0 and (q^*, k) is a remaining packet pair after transmission. New arrivals form common packet pairs. Note that all packet pairs are common packet pairs at the beginning of slot T_0, i.e., $k^* = k, q^* = q$, and under policy p, node n always transmits the earliest deadline packets, i.e., $d_q \le d_k$. This means $(k^*, q), (q^*, k)$ is a sequence of packet pairs so that $d_{q^*} \le d_q, d_{k^*} \le d_k$ where (k^*, q) is at the parent node of node n and (q^*, k) is at node n. Furthermore, (q^*, k) satis es

$d_{q^*} = d_q \le d_k$. Therefore, it is easy to see that all packet pairs satisfy hypothesis 3) by repeating this argument for all transmitted and remaining packet pairs. Suppose a packet k at any node n is expiring under policy p at the end of slot T_0, then there is another packet k^* with $d_{k^*} \le d_k$ and k^* is at node n or its descendent node by hypothesis 3), i.e., k^* is also expiring which contradicting the optimality of policy p^*. Therefore, there is no packet expiring under p at the end of slot T_0, i.e., hypothesis 1) holds. Note that the total number of transmitted packets and received packets are the same under both policies in the uplink tree network and no packets expire under both policies, then the total number of packets at any node under both policies are the same at the end of slot T_0, i.e., hypothesis 2) holds.

Till now we have shown the base case and we need to show that hypothesis 1)-3) hold for T. Similarly, at the beginning of slot T, all nodes have the same amount of packets under p^* and p. The packet pair reforming rule is the same as described in the base case. Consider an arbitrary node n and suppose the capacity of the link between node n and its parent node is 1 without loss of generality. If the link capacity is c, then repeat the same arguments c times. Let k^* and q denote the transmitted packets by node n under policy p^* and p, respectively. Without loss of generality, we assume (k^*, k) and (q^*, q) form different packet pairs at the beginning of slot T, i.e., the end of slot $T-1$. Consider any packet pair (k_0^*, k_0) (assume this packet pair is at an arbitrary node m) in the network and by hypothesis 3) for slot $T-1$, (k_0^*, k_0) has a sequence of packet pairs $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ at node m and its descendent nodes at the beginning of slot T so that $d_{k_1^*} \le d_{k_0}, \ldots, d_{k_i^*} \le d_{k_{i-1}}, d_{k_0^*} \le d_{k_i}$ if $i \ge 1$ and $d_{k_0^*} \le d_{k_0}$ if $i = 0$. We have the following cases:
I) The sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ does not contain the packet pairs (k^*, k) and (q^*, q), then the reforming of (k^*, k) and (q^*, q) and the transmission of (k^*, q) do not in uent the packet pair (k_0^*, k_0);
II) The sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ contains the packet pair (k^*, k) but does not contain (q^*, q). This means node n is node m or a descendent node of node m. By hypothesis 3) for packet pair (q^*, q) in slot $T-1$, there is a sequence $(q^*, q), \ldots, (q_j^*, q_j)$ at node n and its descendent nodes so that $d_{q_1^*} \le d_q, \ldots, d_{q_j^*} \le d_{q_{j-1}}, d_{q^*} \le d_{q_j}$. Without loss of generality, let $(k_0^*, k_0), \ldots, (k^*, k), \ldots, (k_i^*, k_i)$ denote the sequence of (k_0^*, k_0) that contains (k^*, k), then $(k_0^*, k_0), \ldots, (k^*, q), (q_1^*, q_1), \ldots, (q_j^*, q_j), (q^*, k), \ldots, (k_i^*, k_i)$ is a sequence of packet pairs at node m and its descendent nodes so that $d_{k_1^*} \le d_{k_0}, \ldots, d_{q_1^*} \le d_q, \ldots, d_{q^*} \le d_{q_j}, \ldots, d_{k_i^*} \le d_{k_{i-1}}, d_{k_0^*} \le d_{k_i}$. If node n is a descendent node of node m, then after the transmission of packet pair (k^*, q), hypothesis 3) holds for (k_0^*, k_0). If node n is node m, then packets k_0 and q are both at node n at the beginning of slot T, and $d_q \le d_{k_0^*}$ by EDF. Then, $d_{q_1^*} \le d_q \le d_{k_0^*}$ and $(k_0^*, k_0), (q_1^*, q_1), \ldots, (q_j^*, q_j), (q^*, k), \ldots, (k_i^*, k_i)$ is a sequence of packet pairs at node n and its descendent nodes that satis es hypothesis 3) after transmission of (k^*, q). Similarly, if the sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ contains the packet pair (q^*, q) but does not contain (k^*, k), hypothesis 3) also holds for (k_0^*, k_0) at the end of slot T;
III) The sequence $(k_0^*, k_0), \ldots, (k_i^*, k_i)$ contains the packet pairs (k^*, k) and (q^*, q). This means node n is node m or a descendent node of node m. Without loss of generality, let $(k_0^*, k_0), \ldots, (k^*, k), \ldots, (q^*, q), \ldots, (k_i^*, k_i)$ denote the sequence of (k_0^*, k_0) that contains (k^*, k) and (q^*, q). After packet reforming, $(k_0^*, k_0), \ldots, (k^*, q), (q_1^*, q_1), \ldots, (k_i^*, k_i)$ is a sequence of packet pairs at node m and its descendent nodes so that $d_{k_1^*} \le d_{k_0}, \ldots, d_{q_1^*} \le d_q, \ldots, d_{k_i^*} \le d_{k_{i-1}}, d_{k_0^*} \le d_{k_i}$. If node n is a descendent node of node m, then after the transmission of packet pair (k^*, q), hypothesis 3) holds for (k_0^*, k_0). If node n is node m, then packets k_0 and q are both at node n at the beginning of slot T, and $d_q \le d_{k_0^*}$ by the earliest deadline policy. Then, $d_{q_1^*} \le d_q \le d_{k_0^*}$ and $(k_0^*, k_0), (q_1^*, q_1), \ldots, (k_i^*, k_i)$ is a sequence of packet pairs at node n and its descendent nodes that satis es hypothesis 3) after transmission of (k^*, q). Similarly, if $(k_0^*, k_0), \ldots, (q^*, q), \ldots, (k^*, k), \ldots, (k_i^*, k_i)$ is the sequence of (k_0^*, k_0) at the beginning of slot T, hypothesis 3) also holds for (k_0^*, k_0) at the end of slot T with the sequence $(k_0^*, k_0), \ldots, (q^*, k), \ldots, (k_i^*, k_i)$.

Therefore, by repeating the above argument for all transmitted packet pairs, hypothesis 3) holds for all packet pairs at the end of slot T. Suppose a packet k at any node n is expiring under policy p at the end of slot T, then there is another packet k^* with $d_{k^*} \le d_k$ and k^* is at node n or its descendent node by hypothesis 3), i.e., k^* is also expiring which contradicting the optimality of policy p^*. Therefore, there is no packet expiring under p at the end of slot T, i.e., hypothesis 1) holds. Note that the total number of transmitted packets and received packets are the same under both policies in the uplink tree network and no packets expire under both policies, then the total number of packets at any node under both policies are the same at the end of slot T, i.e., hypothesis 2) holds.

Appendix C
Examples

Example 2: Consider a down-link tree (ow destinations are leaves and source nodes can be any other nodes in the tree) shown in Fig. 7 (a) and suppose that the link capacity of each link is 1. Initially at node 4, there are two packets k_1 and k_2 with deadlines $d_{k_1} = 3, d_{k_2} = 3$ whose destinations are nodes 1 and 2, respectively. Suppose that node 4 transmits k_1 to node 3 in time slot 1, and that there is no arrival by the end of slot 1, then node 3 transmits k_1 to node 1 and node 4 transmits k_2 to node 3 in slot 2. Let the adversary inject a packet k_3 with deadline $d_{k_3} = 3$ (whose destination is node 2), at node 3 by the end of slot 2, then by the end of slot 3, either k_2 or k_3 expires. However, this arrival sample is feasible since the off-line algorithm transmits k_2 in slot 1 and all three packets are able to reach their destinations before their deadlines. Now suppose node 4 transmits k_2 to node 3 in time slot 1, and there is no arrival by the end of slot 1, then node 3 transmits

k_2 to node 2 and node 4 transmits k_1 to node 3 in slot 2. Let the adversary inject a packet k_3 with deadline $d_{k_3} = 3$ (whose destination is node 1), at node 3 by the end of slot 2, then by the end of slot 3, either k_1 or k_3 expires. However, this arrival sample is also feasible since the off-line algorithm transmits k_1 in slot 1 and all three packets are able to reach their destinations before their deadlines. This means under scenario (a), no matter what online decision node 4 makes in slot 1, the adversary can always choose future arrivals so that the online decision is worse than the optimal off-line algorithm even though the whole arrival sample is under-loaded. This implies that there is no online algorithm for topology (a) that has the same performance as the optimal off-line algorithm.

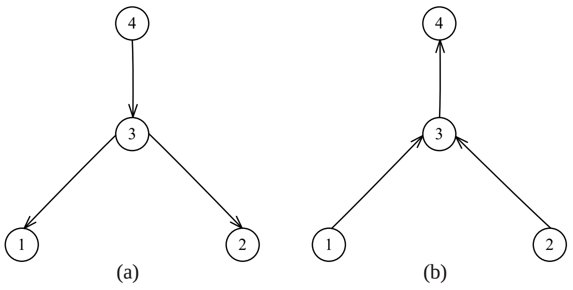


Fig. 7. Example Topologies

Example 3: Consider an uplink tree (all packets are destined to the root) as shown in Fig. 7 (b) and suppose the capacity of link $3 \to 4$ is 2, and the capacities of link $1 \to 3, 2 \to 3$ are 1. Initially, there are two packets k_1 and k_2 with deadline $d_{k_1} = 2, d_{k_2} = 3$ and one packet $k_3 = 2$ at node 2. Suppose node 1 transmits k_1 and node 2 transmits k_3 to node 3 in time slot 1. Let the adversary inject a packet k_4 with deadline $d_{k_4} = 2$ at node 3 and a packet k_5 with $d_{k_5} = 3$ at node 1 by the end of slot 1, then one packet among k_1, k_3, k_4 expires by the end of slot 2 and one packet in k_2, k_5 expires by the end of slot 3, i.e., at least two packets expire. However, only one packet expires for the same arrival sample if node 1 transmits k_2 in slot 1. Similarly, if node 1 transmits k_2 and node 2 transmits k_3 to node 3 in time slot 1, then the adversary does not inject any future arrivals and k_1 expires by the end of slot 2. However, there is no packet expiring if node 1 transmits k_1 in slot 1. Therefore, there is no online algorithm for scenario (b) that has the same performance as the optimal off-line algorithm when the arrival sample can be over-loaded.